Supplementary Information: Supplementary Figures

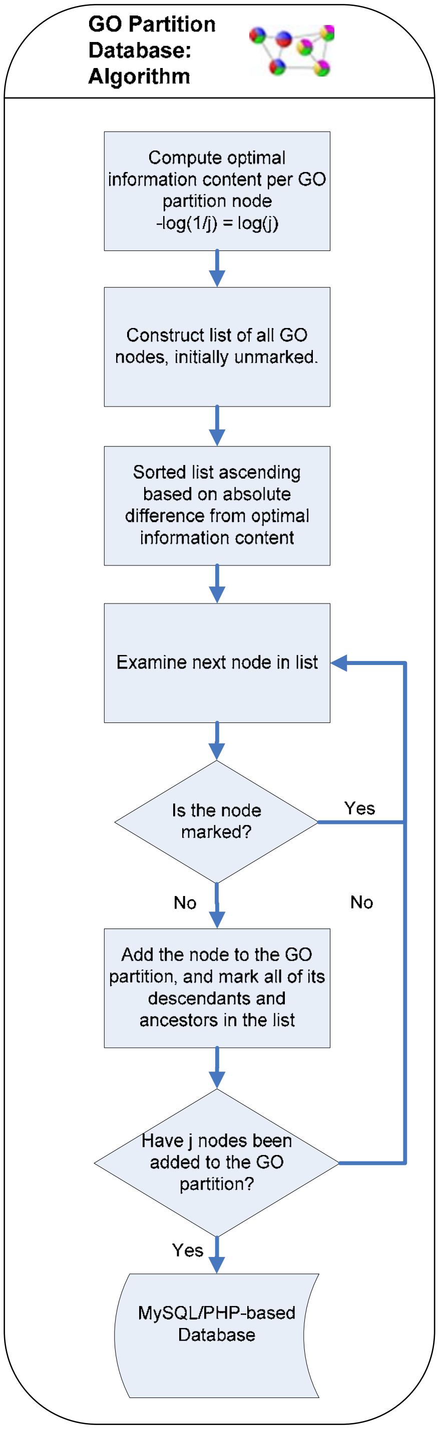
GO Partition Database: Algorithm
Compute optimal information content per GO partition node -log(1/j) = log(j)
Construct list of all GO nodes, initially unmarked.
Sorted list ascending based on absolute difference from optimal information content
Examine next node in list
Is the node marked?
Yes
No
No
Add the node to the GO partition, and mark all of its descendants and ancestors in the list
Have j nodes been added to the GO partition?
Yes
MySQL/PHP-based Database

Figure S1. Gene Ontology Partition Database is based on data obtained from information theoretic analysis of GO. First, the GO node with actual information content closest to the optimal information content (assuming j nodes to select) is chosen. The optimal information content is the self-information (surprisal) of a GO node assuming it has exactly 1/j annotation frequency in the genome at large (see prior equation for self-information). Next, all ancestors and descendants of the GO node just added are “marked” so as not to be added later. The GO node with actual information content that is next closest to the optimal information content is then examined. If the GO node is ”marked,” then the assumption of independence of annotation by a GO partition node is violated; thus, it is discarded, and the GO node with information content that is next closest to optimal is examined. Otherwise, the GO node is kept, and its ancestors and descendants are marked. This process is continued until all j nodes have been chosen as the j GO partition nodes, which collectively make up a GO partition of j nodes. The optimal information content per node for a set of n nodes is defined using an inverse relation: a gene chosen at random would be expected to be annotated by one node from the set of n nodes. To construct the online GO Partition Database, j ranged from 1 to 100, and this algorithm was run for 11 organisms.